PN - JP7064766 A 19950310

PD - 1995-03-10

PR - JP19930208089 19930824

OPD - 1993-08-24

TI - MAXIMUM AND MINIMUM VALUE CALCULATING METHOD FOR PARALLEL COMPUTER

IN - SHINDO TATSUYA

PA - FUJITSU LTD

IC - G06F7/02 ; G06F15/16

PN - JP7064766 A 19950310

PD - 1995-03-10

AP - JP19930208089 19930824

IN - SHINDO TATSUYA

PA - FUJITSU LTD

TI - MAXIMUM AND MINIMUM VALUE CALCULATING METHOD FOR PARALLEL COMPUTER

AB - PURPOSE:To calculate maximum and minimum values in the parallel computer which consciously handles not only maximum and minimum values but also the value of an index to obtain the same result as a sequential computer.

- CONSTITUTION:Values of array elements as objects are compared with each other; and if they are equal to each other, index values are compared with each other, and the smaller index value is taken. That is, stages L1 to L9 where portions in charge of respective parts are individually calculated and a tournament processing is called and tournament processings (function global) L10 to L14 are equal to conventional procedure, but processings L29 to L30 executed in the case of equal inputs in functions to be subjected to maximum value comparison are different from conventional procedure. The minimum value is obtained in the same manner. Another way to take the larger index value in the case of equal inputs can be easily selected. This method is easily transformed that a parameter designates which of the larger index value and the smaller index value should be taken in the case of equal input values.

I - G06F7/02 ;G06F15/16

[Claim(s)]
[Claim 1] They are the maximum and the minimum value operation method characterized by comparing an index value when those values are equal, and taking an index value with the smaller value in the operation which calculates the maximum of an array element or the minimum value, and the index value that shows the position of the array element by the tournament method in a parallel computer while comparing the value of the target array element.
[Claim 2] The maximum and the minimum value operation method according to claim 1 characterized by comparing an index value and taking an index value with the larger value.
[Claim 3] The maximum and the minimum value operation method according to claim 1 or 2 characterized by comparing an index value and determining whether to take an index value with the larger value, or take the index value of the smaller one with the parameter defined beforehand.
[Claim 4] the case of a multidimensional array -- an index array element value -- REKISHIKO -- the claim 1 characterized by comparing with graphical order, or the maximum and the minimum value operation method according to claim 3.

[Detailed Description of the Invention]
[0001]
[Industrial Application] The operation which asks for the index which shows the position of the maximum of an array element or the minimum value, and its element in numerical calculation is often used.
[0002] There is a parallel computer as one of the computers for a high-speed operation. this invention relates to the method of calculating the maximum and the minimum value which used the parallel computer.
[0003]
[Description of the Prior Art] The tournament method is learned as a method of calculating maximum or the minimum value at high speed using a parallel computer. In a parallel computer, the target numerical group (array element) is distributed to each processor, maximum (minimum value) is calculated, respectively, and the maximum (minimum value) between the maximums (minimum value) which after each processor has is calculated. Naturally as for the stage of calculating the maximum (minimum value) of a value to which each processor was distributed, respectively, processing is performed in parallel. A tournament method is taken in order that even the stage of calculating the maximum (minimum value) between [whole] the maximums (minimum value) in each subsequent processor may operate two or more processors in parallel as much as possible.
[0004] Drawing 3 is composition explanatory drawing of a parallel computer. Many processor elements PE are put in order in parallel, and the meantime is combined in Network n. Drawing is the example which combined the processor element (it is described as a processor below) of 8x8 in the two-dimensional anchor ring network. Drawing 4 is drawing showing the concept of the operation of a tournament method performed in such a parallel computer. each processor (the value which it has) -- every two -- a group -- carrying out -- one processor -- one value -- from other processors -- transmitting -- I have you -- those -- more -- size -- a value (or -- more -- smallness -- a value) is calculated The operation .. is repeated and, finally it obtains as maximum (or minimum value) by making every two of the results into a group again, and making one value into a solution. In drawing 4 (A), a processor 0, 1, 2, 3 and 4, 5, 6 and 7, and ... become a group to the 1st time, processors 1, 3, 5, and 7 and ... send data to processors 0, 2, 4, and 6 and ..., and it calculates there, respectively (** shows). next, a processor 0, and 2, 4, 6 and ... a group -- becoming -- processors 2 and 6 ...

processors 0 and 4 -- data are sent to ... and it calculates there, respectively (** shows) Next, processors 0 and 4 and ... become a group, ... (** shows) is repeated, and, finally the maximum as an end result is obtained by the processor 0.

[0005] The example program which described the above tournament processing as a function by false C to drawing 4 (B) is shown. At this example, the number of processors is 2n. It is an individual and is Variable pid. Number respectively peculiar to a processor (equipment item number) It shall be set up. Each processor is myData and myIndex about the value set as the object of tournament processing, and its index. When it sets up and this program is performed, respectively, finally it is pid=0. myData and myIndex of a processor of watch A result is obtained. In addition, send is a function which transmits a message, and the 2nd and the 3rd argument are sent to the processor shown by the 1st argument, and recv is a function which receives a message and receives the message sent to itself by send. An operation function is a portion which calls the function of operations, such as "comparison of maximum", and "comparison of the minimum value."

[0006] When performing a maximum operation etc., a processor makes the index which shows the position on the array of the element accompany an element, and treats it. That is, 2 sets of elements are compared and the index value is made to accompany with the value of a result. Therefore, the index value which accompanies with the value of the direction (direction which won) which remained as a result of comparison also remains undefeated, and the maximum (or minimum value) which finally remained, and the index value which accompanies it are acquired as a solution. The processor which is which position of an array of the maximum (or minimum value), or had it by this understands either, and serves as an index for a subsequent operation.

[0007] By the way, when two values to compare are equal, he is seldom conscious of which value is taken as an index value. If maximum or the minimum value is right, the index value itself is because [2 **-].

[0008] A mold maximum program and the conventional maximum calculation program are serially shown in drawing 2 . The serial type program of (A) investigates the element of Array a from 1 to N, and it is the program from which acquire the biggest value to x and it acquires the index value at that time as a result to i, and is serially written to computers.

[0009] (B) writes the same processing to parallel computers. First, it asks for maximum and its index about the element of a which it takes charge of within each processor (L1-L8). Next, the function global (L9) which performs tournament processing is called, and it asks for the further greatest thing in the maximum which each processor calculated, and its index. the variable which Function global is shown in drawing 4 (B), and the 1st argument x gives the data for comparison here, the variable which the 2nd argument i gives the index value, and the 3rd -- the variable which returns as a result the maximum (minimum value) of x to which each processor set argument &x, and 4th argument &i are variables returned as an index which accompanies it All processors perform this.

[0010] The value of x from which a mold and a parallel connected type are serially obtained as a result is always the same. However, the values i of the index at that time may differ. It is the case where two or more what have x [the same] which each processor sets up as the first argument of Function global exists. By the program of a mold, the smallest thing of an index value is serially adopted as an end result by the property of conditional branching shown in L3 of drawing 2 (A). On the other hand, in the program of a parallel connected type, since the parallel processing by the tournament method is performing size comparison between processors, it is not guaranteed depending on how to assign the element to a processor that the thing of the smallest index value is adopted. Consequently, a different result may be obtained when the program written to be also a basis to sequential processing is changed to

parallel computers by such method. Moreover, since the data assigned to each processor change when the parallelized program is performed with the composition of the different number of processors, results may differ.

[0011]

[Problem(s) to be Solved by the Invention] When the program developed for [usual] serial computers is parallelized to parallel computers, an index value may differ from the result serially obtained by the computer. Moreover, when the number of processors differs in a parallel computer, the index values of a result may differ. Thereby, final results may differ delicately. Such a situation makes difficult the judgment of whether the program is operating correctly or to have parallelized correctly. That is, there is a problem of the compatibility of a result.

[0012] this invention aims at realizing the maximum and the minimum value operation method in the parallel computer which enabled it to obtain the completely same result as a mold serially by he being conscious of it and dealing with maximum/not only minimum value but the value of an index.

[0013]

[Means for Solving the Problem] Drawing 1 shows the program of the example of this invention. Drawing L20-L35 Function which compares shown maximum L29-L32 Processing which was [like] conscious of the size relation of an index value is performed.

[0014] While invention indicated to the claim 1 compares the value of the target array element in the operation which calculates the maximum of an array element or the minimum value, and the index value that shows the position of the array element by the tournament method, when those values are equal, an index value is compared, and an index value with the smaller value is taken.

[0015] In the aforementioned operation, it is the case which was indicated to the claim 2 where compare an index value and an index value with the larger value is taken. In the operation of front 2 term, it is the case which was indicated to the claim 3 where the parameter defined beforehand determines whether an index value with the larger value is taken by comparing an index value, or the index value of the smaller one is taken.

[0016] When the candidate for an operation is a multidimensional array, an index serves as one-dimensional array with the number of elements of the number of dimension of the multidimensional array for an operation, respectively. it indicated to the claim 4 -- as -- case the comparison value for an operation is equal -- those index array element values -- REKUSHIKO -- graphical (it is also called a lexical order) -- it solves by comparing

[0017]

[Function] By making it indicate to a claim 1 or 2, even if the number of the processor of a parallel computer changes, it can guarantee that a result is always the same.

[0018] When parallelizing a sequential program by making it indicate to a claim 3, how to take an index value by the criteria of the loop of a basis can be used properly. For example, what is necessary is just to set up in the example of drawing 2 (A), so that the one where an index value is smaller may be taken since a criteria is if (a[j] <= x) goto L100;. Temporarily, if the criteria is if (a[j] < x) goto L100;, it should just set up so that the one where an index value is larger may be taken. It can respond by the oak from which it carries out also in the above example, and the loop has become descending order, and setting up so that the one where an index value is larger, and the smaller one may be taken conversely, respectively.

[0019] since an index serves as one-dimensional array when making a multidimensional array applicable to an operation -- the comparison -- REKUSHIKO -- it is the same if the point performed graphically is removed When calculating maximum and a minimum value operation by the parallel computer by doing in this way, an index value can be set to a

meaning. Therefore, the compatibility of the result which is not depended on the number of processors and the compatibility of a serial type result can be guaranteed.

[0020]

[Example] Hereafter, the example of this invention is explained with reference to a drawing. Drawing 1 shows the sub routine which calculates maximum as one example of this invention. When the value of an input is equal, it is the example which takes the one where an index value is smaller.

[0021] L1 -L9 are a stage until it calculates a taken charge part individually and calls tournament processing, and L10-L14. The tournament processing (function global) itself is the same as the former. The processing shown in L29-L32 in the function which compares maximum when an input is equal is the important section of the method of this invention.

[0022] When calculating the minimum value, it cannot be overemphasized that what is necessary is just to make it the same. When the value of an input is equal, it can also make it easy to take the one where an index value is larger. Moreover, when the value of an input is equal, it cannot be overemphasized that the deformation which takes the one where an index value is larger, takes small law, or is specified with a parameter is also easy.

[0023] The arrays for an operation may be not only one dimension but many dimensions. in this case -- since an index also serves as an array -- the arrays of an index -- REKISHIKO -- a size relation is judged as compared with graphical order

[0024] In addition, an array element does not need to be a numeric value and may be a character string. this case -- REKISHIKO -- what is necessary is just to judge a size relation as compared with graphical order

[0025]

[Effect of the Invention] As explained above, when the operation which calculates maximum and the minimum value is parallelized according to this invention, the compatibility of a solution with a mold program can be guaranteed serially. Moreover, when performing the parallelized program, it can guarantee that it is not based on the number of processors and a solution always does not change. Such a property is important when guaranteeing that the parallelized program did not spoil the meaning of the program of a basis, but was changed correctly.

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平7－64766

(43)公開日　平成７年(1995)３月10日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　7/02		M		
15/16	３９０	Ｔ　7429－5Ｌ		

審査請求　未請求　請求項の数４　ＯＬ　（全 6 頁）

(21)出願番号　特願平5－208089

(22)出願日　平成５年(1993)８月24日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中1015番地

(72)発明者　進藤　達也
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

(74)代理人　弁理士　井桁　貞一

(54)【発明の名称】　並列計算機における最大・最小値演算方法

(57)【要約】

【目的】並列計算機を用いた最大値・最小値を演算する方法に関し、逐次型と全く同じ結果を得ることができるようにすることを目的とする。

【構成】配列の要素の最大値または最小値と、その配列要素の位置を示すインデックス値とをトーナメント方式で求める演算において、対象とする配列要素の値を比較すると共に、それらの値が等しいときはインデックス値を比較して、その値の小さい方のインデックス値をとるようにする。

実施例のプログラム

```
        x = 0 ;                                          L1
for ( j=担当分下限; j<= 担当分上限; j+= 刻み幅 ) {          L2
    if ( a[j] <= x ) goto L100;                          L3
    x = a[j];                                            L4
    i = j;                                               L5
L100:                                                    L6
    ;                                                    L7
}                                                        L8
global(x, i, &x, &i);                                    L9

global(myData, myIndex, &myData, &myIndex) {             L10
|  図4（B）に示す関数（トーナメント処理）                  .
|                                                        .
}                                                        L14

最大値の比較（入力1，入力2，インデックス1，インデックス2，
            値の出力，インデックスの出力）{               L20
    if（入力1＞入力2）{                                  L21
        値の出力＝入力1；                                L22
        インデックスの出力＝インデックス1；              L23
    } else if（入力1＜入力2）{                           L24
        値の出力＝入力2；                                L25
        インデックスの出力＝インデックス2；              L26
    } else {                                             L27
        値の出力＝入力1；                                L28
        if（インデックス1＜インデックス2）{              L29
        インデックスの出力＝インデックス1；              L30
        } else {                                         L31
        インデックスの出力＝インデックス2；              L32
        }                                                L33
    }                                                    L34
}                                                        L35
```

1

【特許請求の範囲】

【請求項１】　並列計算機において、配列の要素の最大値または最小値と、その配列要素の位置を示すインデックス値とをトーナメント方式で求める演算において、

対象とする配列要素の値を比較すると共に、それらの値が等しいときはインデックス値を比較して、その値の小さい方のインデックス値をとることを特徴とする最大・最小値演算方法。

【請求項２】　インデックス値を比較して、その値の大きい方のインデックス値をとることを特徴とする請求項１に記載の最大・最小値演算方法。

【請求項３】　インデックス値を比較して、その値の大きい方のインデックス値をとるか小さい方のインデックス値をとるかを、あらかじめ定めたパラメータにより決定することを特徴とする請求項１または請求項２に記載の最大・最小値演算方法。

【請求項４】　多次元配列の場合にインデックス配列の要素値をレキシコグラフィカルな順に比較することを特徴とする請求項１ないし請求項３に記載の最大・最小値演算方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】数値計算の中で、配列要素の最大値または最小値とその要素の位置を示すインデックスを求める演算がよく使われる。

【０００２】高速演算用の計算機の一つとして並列計算機がある。本発明は、並列計算機を用いた最大値・最小値を演算する方法に関する。

【０００３】

【従来の技術】並列計算機を用いて最大値または最小値を高速に求める方法として、トーナメント方式が知られている。並列計算機では、対象となる数値群（配列の要素）を各プロセサに配分し、それぞれ最大値（最小値）を求め、そのあと各プロセサのもつ最大値（最小値）の間の最大値（最小値）を求める。各プロセサがそれぞれ配分された値の最大値（最小値）を求める段階は当然に並列に処理が行われる。その後の各プロセサにある最大値（最小値）の間の全体の最大値（最小値）を求める段階でもできるだけ複数のプロセサを並列的に動作させるためトーナメント方式をとる。

【０００４】図３は並列計算機の構成説明図である。プロセサエレメントPEを並列に多数並べ、その間をネットワークｎで結合したものである。図は８×８のプロセサエレメント（以下プロセサと記す）を２次元トーラスネットワークで結合した例である。図４はこのような並列計算機において行なうトーナメント方式の演算の概念を示す図である。各プロセサ（そのもつ値）を２つずつ組にして、１つのプロセサで、１つの値を他のプロセサから転送してもらい、それらのより大なる値（またはより小なる値）を求める。その結果を再び２つずつ組にして

2

・・という演算を繰り返し、最終的に１つの値を解として、すなわち最大値（または最小値）として得る。図４（A）では、１回目にはプロセサ０と１、２と３、４と５、６と７、・・・が組になり、プロセサ１，３，５，７，・・・がプロセサ０，２，４，６，・・・にデータを送り、そこでそれぞれ演算を行なう（①で示す）。次にプロセサ０と２、４と６、・・・が組になり、プロセサ２，６・・・がプロセサ０，４・・・にデータを送り、そこでそれぞれ演算を行なう（②で示す）。次にプロセサ０と４、・・・が組になり・・・（③で示す）ということを繰り返し、最後にプロセサ０に最終結果としての最大値が得られる。

【０００５】図４（B）に以上のトーナメント処理を疑似Ｃ言語で関数として記述したプログラム例を示す。この例では、プロセサの数が2^3個であり、変数pid にそれぞれプロセサ固有の番号（機番）が設定されているものとする。各プロセサはトーナメント処理の対象となる値とそのインデックスとをmyDataとmyIndex に設定しておき、それぞれこのプログラムを実行すると最終的にpid=0 番のプロセサのmyDataとmyIndex に結果が得られる。なお、sendはメッセージを転送する関数であり、第１引数で示されるプロセサに第２と第３の引数を送り、recvはメッセージを受信する関数であり、sendで自分に送られてきたメッセージを受ける。演算関数は、「最大値の比較」や「最小値の比較」等の演算の関数を呼び出す部分である。

【０００６】最大値演算等を行なうとき、プロセサは、その要素の配列上の位置を示すインデックスを要素に付随させて扱う。つまり、２組の要素を比較し、結果の値と共にそのインデックス値を付随させる。従って、比較の結果残った方（勝った方）の値と共に付随するインデックス値も勝ち残り、最終的に残った最大値（または最小値）とそれに付随するインデックス値が解として得られる。これにより、その最大値（または最小値）が配列のどの位置であるか、またそれをもっていたプロセサがどれかが分かり、その後の演算のための指標となる。

【０００７】ところで、比較する２つの値が等しい場合にインデックス値としてどちらの値をとるかはあまり意識されていない。最大値または最小値が正しければインデックス値そのものは２義的なものであるからである。

【０００８】図２に逐次型最大値プログラムと従来の最大値計算プログラムとを示す。（A）の逐次型プログラムは、配列ａの要素を１からＮまで調べて、最も大きな値をｘに、そのときのインデックス値をｉに結果として得るプログラムで、逐次計算機用に書かれたものである。

【０００９】（B）は同様の処理を並列計算機用に書いたものである。まず、各プロセサ内で担当するａの要素について最大値とそのインデックスを求める（L1～L8）。次にトーナメント処理をおこなう関数global（L

9) を呼出し、各プロセサが計算した最大値の中でさらに最大のものとそのインデックスとを求める。ここで、関数globalは図4（B）に示したものであり、第1引数xは比較対象のデータを与える変数、第2引数iはそのインデックス値を与える変数、第3引数&xは各プロセサが設定したxの最大値（最小値）を結果として返す変数、第4引数&iはそれに付随するインデックスとして返す変数である。これを全てのプロセサが実行する。

【0010】逐次型も並列型も結果として得られるxの値は常におなじである。しかし、そのときのインデックスの値iは異なる場合がある。各プロセサが関数globalの第一引数として設定するxが同じものが複数存在する場合である。逐次型のプログラムでは図2（A）のL3に示す条件分岐の性質により、インデックス値の最も小さいものが最終結果として採用される。一方、並列型のプログラムではプロセサ間の大小比較をトーナメント方式による並列処理によって行っているため、プロセサへの要素の割り付け方により、最も小さなインデックス値のものが採用されることは保証されない。その結果、もともとは逐次処理用に書かれたプログラムを、このような方法で並列計算機用に直した場合に、異なる結果が得られる場合がある。また並列化したプログラムを異なるプロセサ数の構成で実行した場合に、各プロセサに割りつけられるデータが変化するため、結果が異なる場合があり得る。

【0011】

【発明が解決しようとする課題】通常の逐次計算機用に開発されたプログラムを並列計算機用に並列化した場合に、逐次計算機で得た結果とは、インデックス値が異なることがある。また、並列計算機においてプロセサの数が異なると、結果のインデックス値が異なることがある。それにより最終的な結果が微妙に異なることがある。このような状況は、プログラムが正しく動作しているか、あるいは正しく並列化されたかの判定を困難にする。すなわち、結果の互換性の問題がある。

【0012】本発明は、最大値／最小値だけでなく、インデックスの値も意識して取り扱うことにより、逐次型と全く同じ結果を得ることができるようにした、並列計算機での最大・最小値演算方法を実現することを目的としている。

【0013】

【課題を解決するための手段】図1は本発明の実施例のプログラムを示すものである。図の L20～L35 に示す最大値の比較を行なう関数の L29～L32 のようにインデックス値の大小関係を意識した処理を行なう。

【0014】請求項1に記載した発明は、配列の要素の最大値または最小値と、その配列要素の位置を示すインデックス値とをトーナメント方式で求める演算において、対象とする配列要素の値を比較すると共に、それらの値が等しいときはインデックス値を比較して、その値

の小さい方のインデックス値をとるようにする。

【0015】請求項2に記載したものは、前記の演算において、インデックス値を比較して、その値の大きい方のインデックス値をとるようにした場合である。請求項3に記載したものは、前2項の演算において、インデックス値を比較して、その値の大きい方のインデックス値をとるか小さい方のインデックス値をとるかを、あらかじめ定めたパラメータにより決定するようにした場合である。

【0016】演算対象が多次元配列の場合には、インデックスはそれぞれ、演算対象の多次元配列の次元数の要素数をもった1次元配列となる。請求項4に記載したように、演算対象の比較値が等しい場合に、それらのインデックス配列の要素値をレクシコグラフィカル（辞書順ともいう）に比較することで解決する。

【0017】

【作用】請求項1又は2に記載したようにすることにより、並列計算機のプロセサの台数が変わっても結果は常に同じであることが保証できる。

【0018】請求項3に記載したようにすることにより、逐次プログラムを並列化する場合に、もとのループの判定条件によりインデックス値のとりかたを使い分けることができる。例えば図2（A）の例では判定条件が

if (a[j] <= x) goto L100;

であるので，インデックス値の小さい方をとるように設定すればよい。仮に判定条件が、

if (a[j] < x) goto L100;

になっていれば、インデックス値の大きい方をとるように設定すればよい。以上の例でもしループが降順になっているなら、それぞれ逆にインデックス値の大きい方、小さい方をとるように設定することで対応できる。

【0019】多次元配列を演算対象とする場合も、インデックスが1次元配列となるのでその比較をレクシコグラフィカルに行なう点を除けば、同様である。このようにすることにより、最大値・最小値演算を並列計算機で演算する場合に、インデックス値を一意に定めることができる。従って、プロセサの数によらない結果の互換性、逐次型との結果の互換性を保証することができる。

【0020】

【実施例】以下、図面を参照して本発明の実施例を説明する。図1は本発明の一実施例として最大値を求めるサブルーチンを示すものである。入力の値が等しい場合はインデックス値の小さい方をとる例である。

【0021】L1 ～L9の、個別に担当分を演算し、トーナメント処理を呼ぶまでの段階と、L10～L14 のトーナメント処理（関数global）そのものは従来と同じである。最大値の比較を行なう関数内のL29～L32に示す、入力が等しい場合の処理が本発明の方法の要部である。

【0022】最小値を求める場合も同様にすればよいことはいうまでもない。入力の値が等しい場合はインデッ

5

クス値の大きい方をとるようにすることも容易にできる。また、入力の値が等しい場合はインデックス値の大きい方をとるか小さい法をとるかパラメータにより指定するようにする変形も容易であることはいうまでもない。

【００２３】演算対象の配列は１次元に限らず多次元であってもよい。その場合はインデックスも配列となるので、インデックスの配列同士をレキシコグラフイカルな順に比較して大小関係を判断する。

【００２４】なお、配列の要素は数値である必要はなく、文字列であってもよい。この場合もレキシコグラフイカルな順に比較して大小関係を判断すればよい。

【００２５】

【発明の効果】以上説明したように、本発明によれば最大値・最小値を求める演算を並列化した場合に逐次型プログラムとの解の互換性を保証することができる。また、並列化したプログラムを実行する場合に、プロセサの数によらず常に解が変わらないことを保証できる。このような性質は、並列化したプログラムがもとのプログラムの意味を損なわず正しく変換されたことを保証する上で重要である。

6

【図面の簡単な説明】

【図１】本発明の一実施例のプログラムの要部を示す図である。

【図２】従来の最大値演算プログラムの例を示す図である。

【図３】並列計算機の構成説明図である。

【図４】トーナメント処理の説明図である。

【符号の説明】

PE　プロセサ（エレメント）　　n　プロセサ間ネットワーク

【図１】

実施例のプログラム

```
                                                    L1
        x＝0；                                      L2
for ( j=担当分下限; j<= 担当分上限; j+= 刻み幅 ) {   L3
    if ( a[j] <= x ) goto L100;                     L4
    x = a[j];                                       L5
    i = j;                                          L6
L100:                                               L7
    ;                                               L8
}                                                   L9
global(x,i,&x,&i);
global(myData,myIndex,&myData,&myIndex)  {          L10
                                                    ・
図４（B）に示す関数（トーナメント処理）            ・
                                                    ・
}                                                   L14


最大値の比較（入力１，入力２，インデックス１，インデックス２，
              値の出力，インデックスの出力）{       L20
    if（入力１＞入力２）{                           L21
        値の出力＝入力１；                          L22
        インデックスの出力＝インデックス１；        L23
    } else if(入力１＜入力２) {                     L24
        値の出力＝入力２；                          L25
        インデックスの出力＝インデックス２；        L26
    } else {                                        L27
        値の出力＝入力１；                          L28
        if（インデックス１＜インデックス２）{       L29
        インデックスの出力＝インデックス１；        L30
        } else {                                    L31
        インデックスの出力＝インデックス２；        L32
        }                                           L33
    }                                               L34
}                                                   L35
```

【図２】

従来の最大値演算プログラムの例

（A）逐次型プログラム

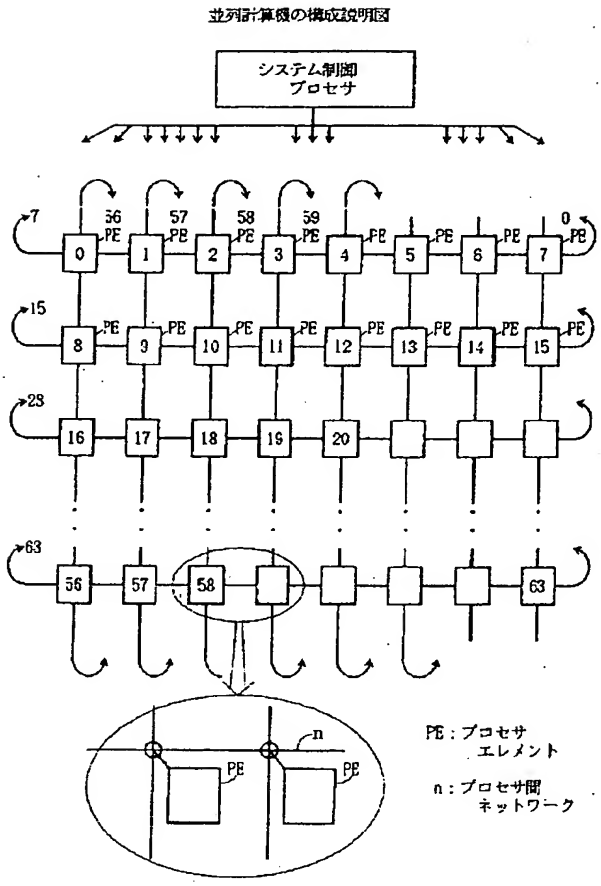
```
        x＝0；                                      L1
for ( j=1; j<=N; j+=1) {                            L2
    if ( a[j] <= x ) goto L100;                     L3
    x = a[j];                                       L4
    i = j;                                          L5
L100:                                               L6
    ;                                               L7
}                                                   L8
```

注：　a[j]　配列　（入力）
　　x　最大値
　　i　そのインデックス

（B）並列型プログラム

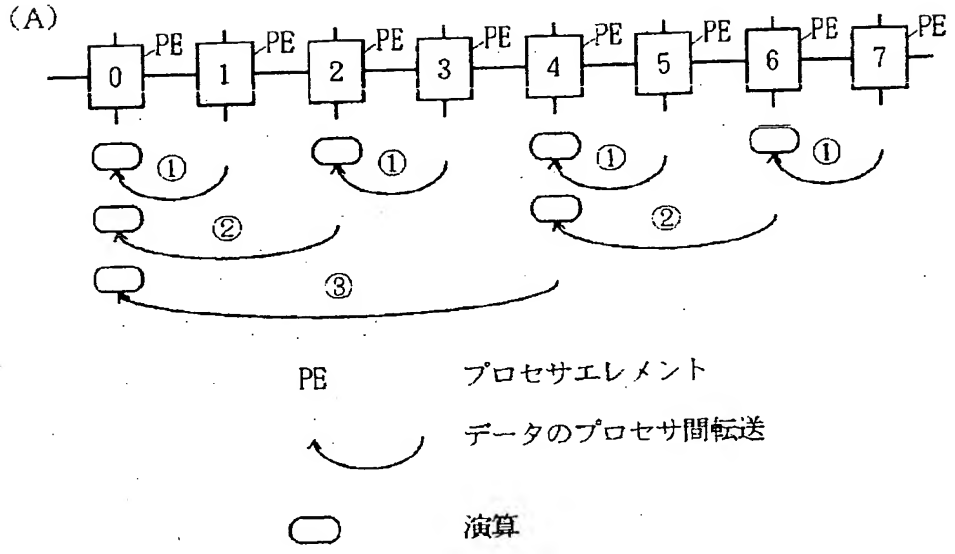
```
                                                    L1
        x＝0；                                      L2
for ( j=担当分下限; j<= 担当分上限; j+= 刻み幅 ) {   L3
    if ( a[j] <= x ) goto L100;                     L4
    x = a[j];                                       L5
    i = j;                                          L6
L100:                                               L7
    ;                                               L8
}                                                   L9
global(x, i, &x, &i);
global(myData,myIndex,&myData,&myIndex)  {          L10
                                                    ・
図４（B）に示す関数（トーナメント処理）            ・
                                                    ・
}                                                   L14


最大値の比較（入力１，入力２，インデックス１，インデックス２，
              値の出力，インデックスの出力）{       L20
    if( 入力１＞入力２）{                           L21
        値の出力＝入力１；                          L22
        インデックスの出力＝インデックス１；        L23
    } else {                                        L24
        値の出力＝入力２；                          L25
        インデックスの出力＝インデックス２；        L26
    }                                               L27
}                                                   L28
```

【図３】



【図４】

トーナメント処理の説明図



（B）上図の処理を疑似C言語の関数として記述したプログラム例

```
global(myData, myIndex, &myData, &myIndex)  {
  sendFlag=0;
  for(i=0; i<n; i++) {
    if(sendFlag==0)  {
      if(pidのi ビット目==1) {
        send(pidの iビット目を 0に置き換えたプロセサID, myData, myIndex);
        sendFlag=1;
      } else { /*  pidの iビット目==0 */
        recv(&rcvData, &rcvIndex);
        演算関数(myData, rcvData, myIndex, rcvIndex, &outData, &outIndex);
        myData=outData;
        myInex=outIndex;
      }
    }
  }
}
```

注：
1. プロセサ数は2^n個、pid は各プロセサの機番を設定する変数
2. myData, myIndexは演算対象となるデータとインデックス
3. sendは第１引数のプロセサに第２第３引数の値を送る関数
4. recvはsendで送られてきた値を受け取る関数
5. 演算関数は最大値・最小値比較等の演算を行なう関数
6. 最終結果はpid=0 のプロセサのmyData, myIndexに残る。